(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 481 097 A1**

(12) **EUROPEAN PATENT APPLICATION**

(21) Application number: **90117819.4**

(51) Int. Cl.5: **G06F 11/26**

(22) Date of filing: **15.09.90**

(43) Date of publication of application:
22.04.92 Bulletin 92/17

(84) Designated Contracting States:
DE FR GB NL SE

(71) Applicant: **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: **Diebold, Ulrich, Dipl.-Ing.**
Gänsbergring 74
W-7033 Herrenberg(DE)
Inventor: **Riegler, Joachim, Dipl.-Ing.**
Weingartenstrasse 9
W-7268 Gechingen(DE)
Inventor: **Rost, Peter, Dipl.-Ing.**
In den Lunktellen
W-7260 Calw(DE)
Inventor: **Schmidt, Manfred, Dipl.-Phys.**
Dachenhäuserweg 37
W-7036 Schönaich(DE)
Inventor: **Torreiter, Otto, Dipl.-Ing.**
Fleinsbachstrasse 14
W-7022 Leinfelden-Echterdingen(DE)
Inventor: **Verwegen, Peter, Dipl.-Ing.**
Hohenstaufenstrasse 11
W-7030 Böblingen(DE)
Inventor: **Weiland, Dawn**
Im Heinental 21
W-7277 Wildberg 4(DE)
Inventor: **Wendel, Dieter, Dipl.-Ing.**
Schöneberger Weg 14
W-7032 Sindelfingen(DE)

(74) Representative: **Jost, Ottokarl, Dipl.-Ing.**
IBM Deutschland GmbH Patentwesen und Urheberrecht Schönaicher Strasse 220
W-7030 Böblingen(DE)

(54) **Method and apparatus for testing a VLSI device.**

(57) A method and apparatus for testing a VLSI device 10 are described. The invention uses the idea that the internal logic of the VLSI device can be broken down into linked sections or cones. Thus, after completing the test of all the logic gates 110, 120 in one cone of the device 10, it will not be necessary to retest those gates 120 which overlap into other cones. Hence, some of the bit values placed on the input latches 30 to the device 10 are irrelevant to the test. The apparatus incorporates a Linear Feedback Shift Register (300) which is fed by a seed to produce a bit pattern to test the VLSI device (10). The seed is so chosen that the LFSR generates the required bit values on the input latches 30 which are required for the particular test being carried out and pseudo-random values for all other latches.

EP 0 481 097 A1

Field of the Invention

The present invention relates to methods of testing complex combinatorial and sequential logic circuits embodied in very large scale (VLSI) devices.

Description of the Prior Art

A fault occurring anywhere in a VLSI device can have its effect propagated through a number of gates before reaching a testable output of the device. Level sensitive scan design (LSSD) rules were devised to eliminate the complications in testing caused by this propagation. These rules, which were first published in an article entitled "A Logic Design Structure for LSI Testability" by E.B.Eichelberger and T.W.Williams in the Proceedings of the 14th Design Automation Conference, pp 462-468, impose a clocked structure on the logic circuit and require all the inputs and outputs of the logic circuits to be tied together to form a series of shift register scan paths. Fig. 1 shows a simplified tester built using these principles. Test unit 40 comprises a memory 42 to store the test pattern data, means 44 to simulate the response to the device under test and means 46 to supply the test pattern to the device under test. Test signals generated in unit 40 can therefore be applied to the device under test 10 using the device inputs 30 which are connected together to form a shift register. The input values are clocked through the shift register from the unit 40 using the connections shown as dotted lines on the Figure. After the test has been carried out by clocking the input values through the logic circuit 15 to be tested, the results appear at the output latches 20 which are also connected together to form a shift register. These output values are clocked out of the shift register into the comparator 50 which compares the obtained results with the simulated expected results and signals if there is a difference which would indicate the presence of a fault in the device under test 10.

The conventional method of generating logic test patterns is known as "Deterministic Stored-Pattern Testing" (DSPT). This involves using a deterministic algorithm to produce patterns which guarantee detection of specified logical faults which include, but are not limited to, stuck-at faults (i.e. faults in which a gate does not change its output in response to an input). Since each test pattern and its expected output response are stored as uncompressed vectors of signal values in the tester, large amounts of memory are required. It has been shown that as circuits become more complex, the number of possible stuck-at faults grows spectacularly as the number of gates in the circuit increase. The implication is that the number of possible test patterns required in order to test the circuit will also greatly increase. Hence the time required to test the circuit will be extended and memory requirements for storing all the test vectors will increase.

In order to reduce the time taken to generate the test patterns and the amount of memory required to store the patterns generated, so-called self-testing methods have been developed. These involve the use of pseudo-random pattern generators and response compression structures actually built into the device. Using such structures eliminates the computer time needed to generate the tests, whilst placing the elements required for testing directly to the device under test. This allows the application of large number of test patterns to the device, in a reasonable length of time. In two articles by Konemann, Mucha and Zwiehoff, "Built-in Logic Block Observation Techniques", 1979 IEEE Test Conference, pp37-40, Cherry Hill, NJ, October 1979 and "Built-in Test for Complex Digital Integrated Circuits", IEEE J.Solid-State Circ, vol SC-15, Nr 3, pp 315-319, June 1980, a modification to a shift register scan path - called the linear feedback shift register (LFSR) - was disclosed which could be used as an input signal generator and/or as an output data compression circuit.

A special self-test architecture is known as the STUMPS method, STUMPS being short for Self-Test using a MISR (multiple-input signature register) and a Parallel SRSG (shift register sequence generator). The basic principles of the method are well known and are described in a number of documents (e.g. EP-B-0 108 256 (IBM), US-A-4 519 078 (KOMONYTSJKY), US-A-713 605 (IYER et al), US-A-4 910 735 (YAMASHITA) or B.I.DERVISOLGLU "Scan-path architecture for pseudo random testing", IEEE Design and Test of Computers, Aug 1989, pp 32-48). It is found that, although STUMPS greatly reduces the amount of test data storage required, the quality of the tests is not high. An article by BASSET et al, "Low-cost testing of high density logic components", IEEE Design and Test of Computers, April 1990, pp 15-27, reports that one can only expect 90% fault coverage from STUMPS for a representative cross-section of chips. A combination of STUMPS and DPST would, it was concluded in the same article, increase the fault coverage. However, in order to determine the last 5 to 10% of faults (the so-called self-test escapes), experience has shown that one must generate 50-70% of the complete set of DPST test patterns.

Instead BASSETT et al favoured the use of the Weighted Random Pattern Test (WRPT) in which the design of the LFSR pattern generator is changed to produce a variable distribution of logical 1's and 0's for each test pattern input bit. This method selectively biases the input latches to a greater probability of 1 or 0

as required. Although fault coverage was much improved over the STUMPS method, substantially more complex hardware was needed to generate the pattern. An article by J. WAICUKAUSKI et al entitled "A method for generating weighted random test patterns" in the IBM Journal of Research and Development, vol 33, nr 2, March 1989, pp149-161, illustrated a number of ways in which WRPT patterns could be generated. The results obtained, however, suggested a fault coverage of 94-99% and thus deterministic pattern testing in the order of 50% of a full stored pattern test will still be required if one wishes to detect 99.9% of all faults.

None of the prior art discloses the use of an LSFR circuit to simultaneously produce input patterns and to compress output results.

Summary of the Invention

It is an object of the invention to provide a method and a test pattern generator which can be used for testing the logic of a device. The means used for generating the test pattern does not require complex hardware nor large amounts of memory space to store the individual test pattern.

The test pattern generator can be directly incorporated onto the chip as well as being built into a stand-alone pattern tester. If incorporated onto a chip, the pattern generator can be further advantageously used for self-testing the defice.

The invention achieves these objects by being made of a simple LFSR, the output of which is fed into a scan path of a series of shift registers forming the inputs of the device to be tested. The LFSR is initially fed with a series of initial values or 'seed' which causes required data values to be clocked into specific input latches of the device under test. A small memory stores all the seeds which are required to ensure that all the gates of the logic device have been tested.

Brief Description of the Drawings

Fig. 1 shows the basic method of testing a device.
Fig. 2 is an illustrative example of a device under test, showing the logic gates connected between the input and output gates.
Fig. 3 shows a simple linear feedback shift register.
Fig. 4 shows an example of the generation of a seed.

Detailed Description of the invention

The basic principle behind the invention can be understood by considering Fig. 2 which shows the internal logic of a typical device under test. The device consists of a number of internal logic gates 110a-f, 120a-c, 130a-h connected between input latches 100a-n and output latches 140a, 140b. It should be noted that this is merely an illustrative example, in practice a device under test would have many more input and output ports and many more logic gates connected between these ports. The input and output latches are separately connected together to each form a scan path represented by the dotted line in the Figure. If we now consider the pathways through the logic circuit of the device leading to the output latches 140 from the input latches 100, it can be seen that the values on the input latches 100a to 100h will affect the responses output on 140a. Similarly, the values on the input latches 100e to 100n will affect the results on output 140b. This means that two 'cones' can be defined, one of which runs from latches 100a to 100h to the output latch 140a and the other of which runs from input latches 100e to 100n to output latch 140b. These cones are marked on the Figure by the dashed lines. It will also be noted that the two cones overlap and that logic gates 120a-c are in both cones. This means that, if all of the logic gates in cone 1 have been tested and shown to have no faults, then in principle only those gates in cone 2 which do not also fall into cone 1 need to be tested in order to complete the test of the circuit (i.e. gates 130a-h). It should also be noted that, when testing cone 2, the values of the bits placed in latches 100e-h cannot be ignored since the output of tested logic gate 120c is connected to the input of untested logic gate 130f. However, the values of the bits placed in latches 100a-d are irrelevant to the test of the latches in cone 2. This means that three different types of bits can be defined which are placed in the latches 100a-n in order to test the logic gates: test-relevant bit positions (labelled R in table 1) which have to take specified values in order to test the cone; support-net bit positions (labelled A in table 1) which must have one of several specific values in order to test a cone and non-relevant bit positions (labelled X in table 1) whose value is not important in testing the cone. Table 1 shows the three different test procedures which are required in order to thoroughly test the logic network of Fig. 2.

Table 1

Net input to Latch 100	cone 1 a b c d	overlap e f g h	cone2 i j k l m n
Procedure A): Test nets common for cone 1 and cone 2			
	A A A A	R R R R	A A A A A A
	.....	.....	.....
Procedure B): Test nets of cone 1 and cone 2			
	R R R R	A A A A	R R R R R R
	.....	.....	.....
Procedure C): Test nets of cone 2			
	X X X X	A A A A	R R R R R R
	.....	.....	.....

Meaning of flags:

- o R: Test-relevant bit positions - no alternative
- o A: Support-net inputs - there are some alternatives
- o X: Non-relevant bit positions - any value

In procedure A only the values contained in latches 100e-h are significant for testing the logic gates 120a-c contained in the overlap of the cones. During the testing procedures all possible combinations of 1's and 0's will be therefore placed on these latches 100e-h so that logic gates 120a-c are tested under all conditions. The bits stored in latches 100a-d and 100i-n are relevant only in that a meaningful output must be obtained at the output gates 140a and 140b. In procedure B, the gates in the individual cones are tested but not those in the overlap (i.e. gates 110a-f and 130a-h but not 120a-c). This means that the values of the bits placed in latches 100a-d and 100i-n are test-relevant and are cycled through all possible 1's and 0's combinations but those in latches 100e-h are only support-net inputs. Finally in Procedure C, only the logic gates 130a-h in cone 2 are tested. Again the bits in latches 100j-n are test-relevant and all possible combinations tested but the bits in latches 100f-h are support-net bits. However, in this case the values of the bits in latches 100a-d are irrelevant to the results of the test. It can therefore be seen that a large number of the test patterns which might be used for testing the logic gates in the device under test are redundant since they will generate no new information about faults in the circuit.

This idea of test-relevant bits can be used in conjunction with an LFSR to generate the test patterns for testing logic circuits. To see how this works, the method of generating patterns using an LFSR must be first reviewed. Fig. 3 shows a simplified LFSR 200, consisting of three latches 200a-c which produces an output 205 which can be clocked through to the scan path 210. The output of latch 200a is connected to both an XOR gate 202 and to the latch 200b. The output of latch 200c is connected both to the output 205 (and hence into the scan path 210) and also to the XOR gate 202. As the data is clocked through the LFSR (meaning that the value of 200a is placed in 200b, the value of 200b is placed in 200c and the value of 200c is transferred to the scan path), the new value of 200a is the result of the XOR operation on the previous value of 200a and the previous value of 200c. By choosing the appropriate initial values for the latches 200a-c in the LFSR 200, we can generate any possible combination of bits in the scan path. Fig. 3 shows how the initial value of 100 loaded into the LFSR 200 generates a series of 0's and 1's through the scan path 210.

Now suppose that, instead of wishing to generate the pseudo-random test pattern as shown in Fig. 3, it is wished to generate a test pattern in which the bits stored in some of the latches 210 of the scan path are fixed. These bits represent the test relevant bit positions which were described above. Fig. 4a shows such a simulated scan path 310 in which the bits stored in latches 310c and 310e are set to be 1 and 0 respectively. The values of the bits stored in the other latches are not relevant. If, for the purposes of

calculation, the values of the latches 300a-c of the LFSR 300 are arbitrarily set to a,b,c respectively, then we can 'clock back' the bits from the simulated scan path 310 into the LFSR 300 to obtain the initial 'seed' values for the LFSR 300 which can then be later used to generate the required test pattern with the test relevant bit positions having their required values. This clocking back procedure, which is carried out as a simulation and does not physically happen on the device under test 10, is illustrated by the various steps marked on Figure 4a.

In the first step the values stored in the latches 310b-g are all moved one latch to the left. So that the value previously stored in latch 310g is now stored in latch 310f, the value in 310f is now stored in 310e and so on. The value stored in latch 310a is ignored since it is a non-test relevant bit and thus latch 300c should take the result of the XOR operation performed by gate 302 on the values previously stored in latches 300a and 300b. In fact since, at this stage, no test relevant bit values have been clocked back from the scan path 310 into the LSFR 300, the value stored in latch 300c is purely arbitrary and it can be defined to be 'a' for simplicity. In the next step, the values stored in latches 310b-g are again moved one latch to the left. The value contained in latch 310a is now a test relevant bit and has the value 1. It is passed to latch 300c of the LSFR 300. The value that should be stored in latch 300c at this step is the result of the XOR operation performed in XOR gate 302, i.e. b + c. Thus we define the result of this XOR operation to be 1. In step 3 and step 4, the values stored in latch 310a are non relevant test bits and hence the value stored in latch 300c is not required to take any value and is therefore the result of the XOR operation of the values stored in latches 300a and 300b. In step 5, however, latch 300c is fed with the test relevant bit 0 from latch 310a. This defines the result of the XOR operation 1 + (a + c) to be 0 since latch 300a previously contained the value of 1 and latch 300b the value a + c. The simulation is continued for two more steps 6 and 7 even though the last two values are non relevant test bits. These steps are, however, necessary to ensure that the LSFR 300, when generating the complete test pattern, will produce bits with known values for all the latches 310a-g of the scan path 310 including the test relevant bits in the correct positions. The result of this operation can be seen from the final step 7, latch 310a must contain a 0, latch 300b will contain the value a and latch 300c will contain the value of a + 1. Since a can take the value either 0 or 1, this means that two seeds (001 or 010) will generate the required test pattern. Fig. 4b demonstrates this for the seed 010.

Of course, in an actual test situation, the scan path would not consist of a mere seven latches of which only two values will be test relevant and thus have their values fixed. However, by increasing the size of the LFSR and by choosing suitable feedback paths for connection to XOR gates, it will be possible to generate any required pattern. Since only the seed for feeding the LFSR needs to be stored, the amount of memory required to implement this test pattern generating system is much smaller than would be required to directly store all the deterministic patterns.

The described test pattern generator can be either implemented directly on the chip to enable rapid self-testing of the device or it can be incorporated into a test apparatus to which the device is attached. In order to save more memory space and to speed up the testing process, the output results from testing can be additionally compressed using commonly known methods such as the multiple input signature register (MISR) described in the article by J.L.CARTER, "Improved signature test for VLSI Circuits", IBM Technical Disclosure Bulletin, vol 26, no 3A, August 1983, pp 965-967.

Claims

1. Apparatus for testing a logic device (10) comprising

means (40) for generating a test pattern for testing the device (10);

device inputs (30) for loading the initial test pattern into the device (10);

device outputs (20) for outputting the results from the device (10);

means (50) for comparing the output results from the device with the results expected and;

means for signalling whether the said logic device has passed the test

characterised in that

said means (40) for generating a test pattern produces specified bit values on certain of the said device

inputs (30) and pseudo-random bit values on others of the said device inputs (30).

2. Apparatus for testing a logic device (10) according to claim 1 further characterised in that

said means (40) for generating a test pattern comprises a linear feedback shift register (300).

3. Apparatus for testing a logic device (10) according to claim 2 further characterised in that

the said linear feedback shift register (300) is fed by pre-calculated initial values (seed).

4. Apparatus for testing a logic device (10) according to claim 1 further characterised in that

the said device inputs (30) are connected together to form a shift register scan path which is further connected to the said means (40) for generating a test pattern for testing the device (10).

5. Apparatus for testing a logic device (10) according to claim 1 further characterised in that

the said device outputs (20) are connected together to form a shift register scan path which is further connected to the means (50) for comparing the output results from the device with the results expected.

6. Apparatus for testing a logic device (10) according to claim 5 further characterised in that

the means (50) for comparing the output results from the device with the results expected includes means to compress the data before carrying out the comparison.

7. Apparatus for testing a logic device (10) according to claim 6 further characterised in that

the said device outputs (20) and the means (50) for comparing the output results from the device with the results expected together form a multiple input signature register.

8. Apparatus for testing a logic device (10) according to any of the above claims characterised in that

the said apparatus is part of a logic device testing machine into which the said logic device (10) is placed.

9. Apparatus for testing a logic device (10) according to claims 1 to 7 characterised in that

the said apparatus is incorporated into the chip on which the said logic device (10) is constructed and is used for self-testing the said logic device (10).

10. Method for testing a logic device (10) comprising

generating a test pattern;

clocking the test pattern through the said logic device (10);

comparing the results output by the chip with the expected results and;

signalling if a difference is found;

characterised in that

the said method of generating the test pattern includes calculating which device inputs (30) are relevant for the desired logic gates and

calculating a test pattern to apply the desired bit values to the said device inputs (30).

11. Method for testing a logic device (10) according to claim 10 further characterised in that

said method for generating a test pattern includes generating a seed for use in a linear feedback shift register (300) to produce the desired test pattern.

12. Method for testing a logic device (10) according to claim 10 further characterised in that

said method for comparing the results output by the chip with the expected results and signalling if a difference is found includes compressing the output results and comparing the said compressed results with compressed said expected results.

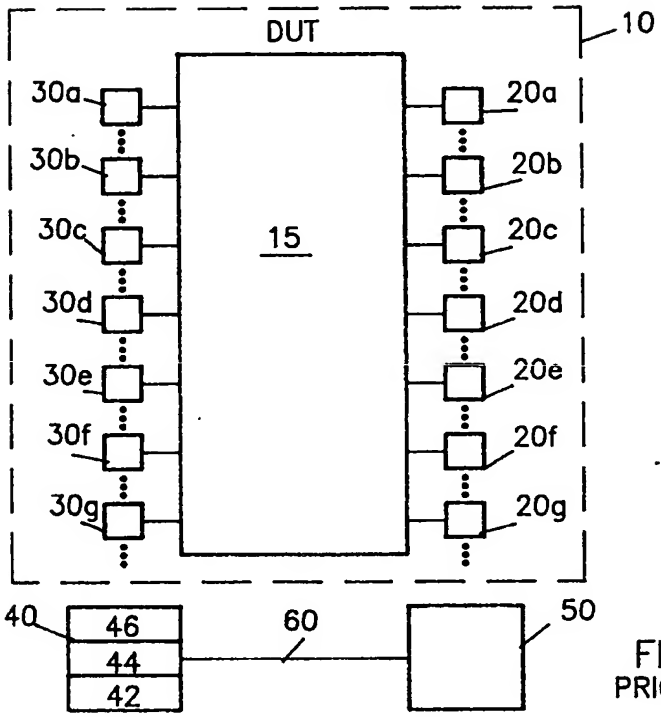


FIG. 1
PRIOR ART

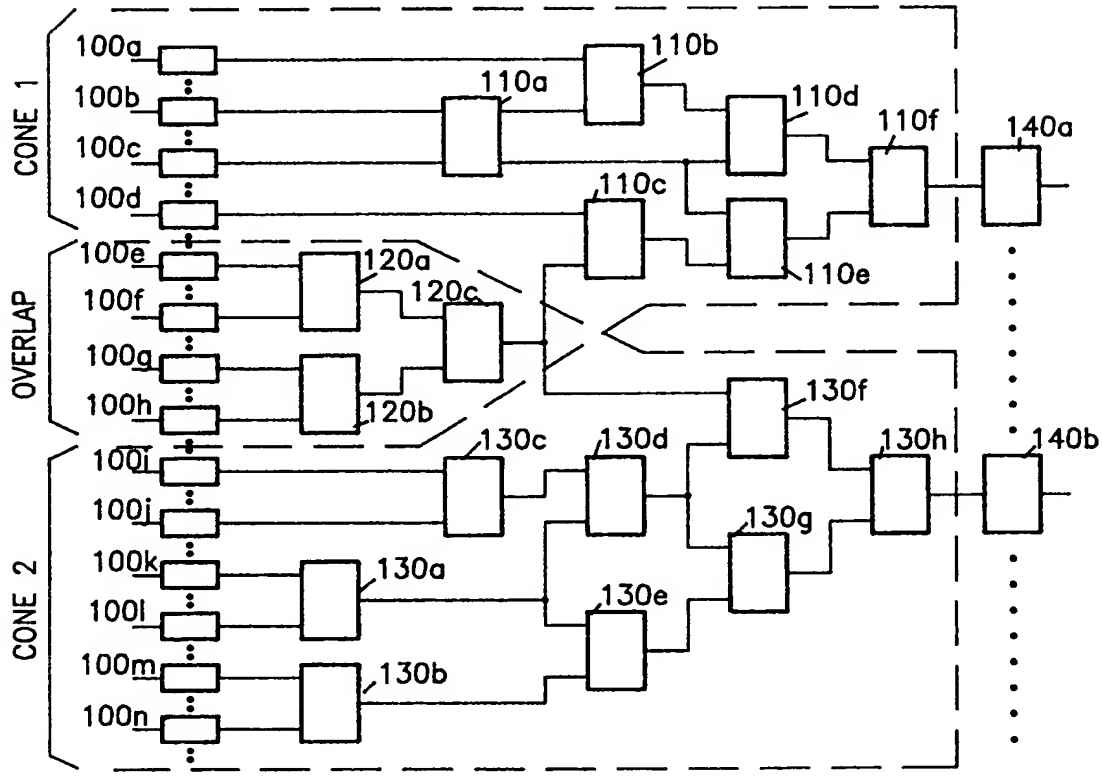
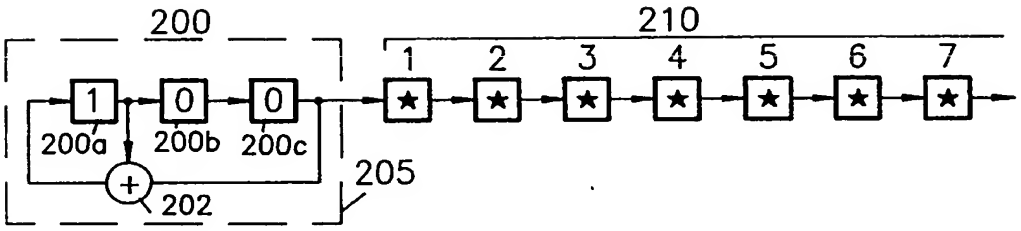


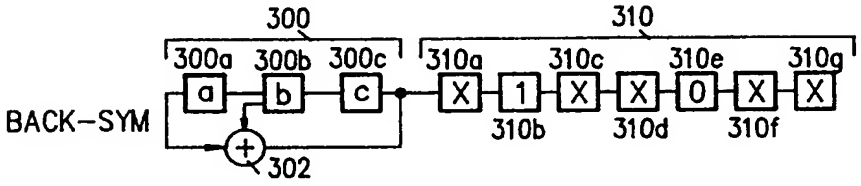
FIG. 2



1	1	0	0	★	★	★	★	★	★
1	1	1	0	0	★	★	★	★	★
0	1	1	1	0	0	★	★	★	★
1	0	1	1	1	0	0	★	★	★
0	1	0	1	1	1	0	0	★	★
0	0	1	0	1	1	1	0	0	★
1	0	0	1	0	1	1	1	0	0
1	1	0	0	1	0	1	1	1	0
1	1	1	0	0	1	0	1	1	1
0	1	1	1	0	0	1	0	1	1
1	0	1	1	1	0	0	1	0	1
0	1	0	1	1	1	0	0	1	0
0	0	1	0	1	1	1	0	0	1
1	0	0	1	0	1	1	1	0	0
1	1	0	0	1	0	1	1	1	0

FIG. 3

START OF SEED CALCULATION



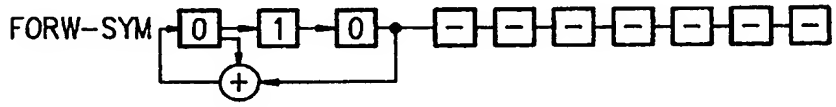
STEP 1	b	c	a	1	x	x	0	x	x	–	
– 2	c	a	1	x	x	0	x	x	–	–	(b+c):=1
– 3	a	1	a+c	x	0	x	x	–	–	–	
– 4	1	a+c	a+1	0	x	x	–	–	–	–	
– 5	1	a+1	0	x	x		–	–	–	–	1+(a+c):=0
– 6	a+1	0	a	x	–	–	–	–	–	–	
– 7	0	a	a+1	–	–	–	–	–	–	–	EMPTY PATTERN

RESULT OF SEED CALCULATION:

SEED 1 =	0	0	1	FOR a:= 0	x = NON TEST RELEVANT BITS
SEED 2 =	0	1	0	a:= 1	1, 0 = TEST RELEVANT BITS

FIG. 4A

START OF PATTERN GENERATION (SEED = 0 1 0)



STEP 1	0	0	1	0	–	–	–	–	–	–	
– 2	1	0	0	1	0	–	–	–	–	–	(b+c):= 1
– 3	1	1	0	0	1	0	–	–	–	–	
– 4	1	1	1	0	0	1	0	–	–	–	
– 5	0	1	1	1	0	0	1	0	–	–	1+(a+c):= 0
– 6	1	0	1	1	1	0	0	1	0	–	
– 7	0	1	0	1	1	1	0	0	1	0	
RESULTING PATTERN:				1	1	1	0	0	1	0	

FIG. 4b

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平4－236378

(43)公開日　平成４年(1992) ８月25日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０１Ｒ　31/318				
		6912－2G	Ｇ０１Ｒ　31／28	Ａ

審査請求　有　　請求項の数12(全 7 頁)

(21)出願番号　特願平3－133342

(22)出願日　平成３年(1991) ５月10日

(31)優先権主張番号　９０１１７８１９．４

(32)優先日　1990年９月15日

(33)優先権主張国　ドイツ（ＤＥ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　ウルリツヒ・デイーボルト
ドイツ連邦共和国　7033　ヘーレンベルク　ゲンシユベルクリング　74

(74)代理人　弁理士　頓宮　孝一　（外４名）

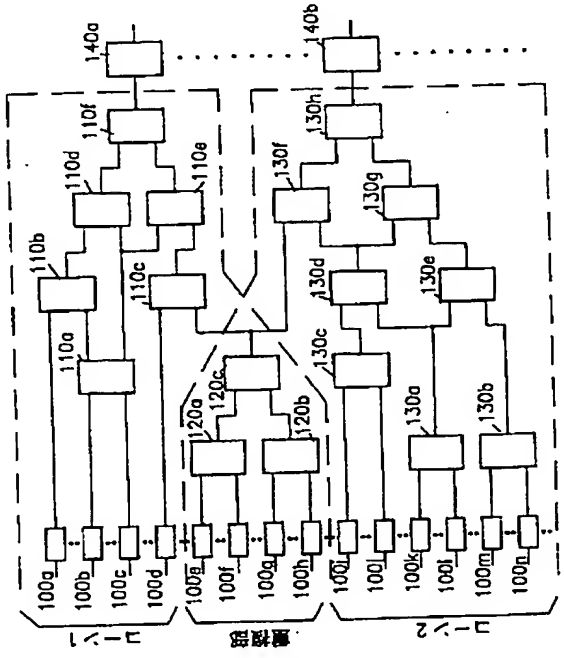
最終頁に続く

(54)【発明の名称】　論理装置を試験する方法および装置

(57)【要約】　（修正有）

【目的】ＶＬＳＩ装置を試験するための方法および装置を提供する。

【構成】ＶＬＳＩ装置の内部論理回路は関連した部分すなわちコーンに分解できるというアイデアを用いる。従って、装置の一つのコーンにおけるすべての論理ゲート１１０，１２０の試験を完了すれば、他のコーンと重複するゲート１２０の再試験は不要である。そのため、装置への入力ラッチに設定されるビット値のいくつかは試験に無関係である。装置は線形フィードバック・シフトレジスタ（ＬＦＳＲ）を備え、これには、ＶＬＳＩ装置を試験するためのビットパターンを生成するシードが入力される。このシードは、特定の試験を行うために必要なビット値を入力ラッチに対してＬＦＳＲが発生し、そして他のすべてのラッチに対して疑似ランダム値を発生するように選択される。



1

【特許請求の範囲】

【請求項１】論理装置（１０）を試験する装置であって、論理装置（１０）を試験するためのテストパターンを発生する手段（４０）と、論理装置（１０）に初期テストパターンをロードする装置入力（３０）と、論理装置（１０）からの結果を出力する装置出力（２０）と、論理装置からの出力結果を期待される結果と比較する手段（５０）と、前記論理装置の試験が終了したかどうかを示す信号を出力する手段とを備えた試験装置において、テストパターンを発生する前記手段（４０）は、前記装置入力（３０）のあるものに対して特定のビット値を生成し、前記装置入力（３０）の他のものに対して疑似ランダムビット値を生成することを特徴とする論理装置（１０）を試験する試験装置。

【請求項２】テストパターンを発生するための前記手段（４０）は、線形フィードバック・シフトレジスタ（３００）を備えたことを特徴とする請求項１記載の論理装置（１０）を試験する試験装置。

【請求項３】予め計算された初期値（シード）が前記線形フィードバック・シフトレジスタ（３００）に入力されることを特徴とする請求項２記載の論理装置を試験する試験装置。

【請求項４】前記装置入力（３０）は共に接続されてシフトレジスタ・スキャンパスを形成し、このシフトレジスタ・スキャンパスはさらに、前記論理装置（１０）を試験するためのテストパターンを発生する前記手段（４０）に接続されていることを特徴とする請求項１記載の論理装置（１０）を試験する試験装置。

【請求項５】前記装置出力（２０）は共に接続されてシフトレジスタ・スキャンパスを形成し、このシフトレジスタ・スキャンパスはさらに、論理装置からの出力結果を期待される結果と比較するための手段（５０）に接続されていることを特徴とする請求項１記載の論理装置（１０）を試験する試験装置。

【請求項６】論理装置からの出力結果を期待される結果と比較するための手段（５０）は、比較を行う前にデータを圧縮する手段を含むことを特徴とする請求項５記載の論理装置（１０）を試験する試験装置。

【請求項７】前記装置出力（２０）および論理装置からの出力結果を期待される結果と比較するための手段（５０）は、共にマルチ入力シグナチャ・レジスタを形成することを特徴とする請求項６記載の論理装置（１０）を試験する試験装置。

【請求項８】前記装置は、前記論理装置（１０）が取り付けられる論理装置試験装置の一部であることを特徴とする請求項１～７のいずれかに記載の論理装置（１０）を試験する試験装置。

【請求項９】前記装置は、前記論理装置（１０）が構成されるチップに組み込まれ、前記論理装置（１０）の自己試験のために用いられることを特徴とする請求項１～

2

７のいずれかに記載の論理装置（１０）を試験する試験装置。

【請求項１０】論理装置（１０）を試験する方法であって、テストパターンを発生し、前記論理装置（１０）を通じてテストパターンをクロックに同期して伝送し、チップによって出力される結果を期待される結果と比較し、差が見つかったかどうかを示す信号を出力する試験方法において、テストパターンを発生する前記方法は、どの装置入力（３０）が所望の論理ゲートに関連しているかを計算することを含み、前記装置入力（３０）に所望のビット値を与えるためにテストパターンを計算することを特徴とする論理装置（１０）を試験する試験方法。

【請求項１１】前記テストパターンを発生するための前記方法は、所望のテストパターンを生成するために、線形フィードバック・シフトレジスタで用いるシードを発生することを含むことを特徴とする請求項１０記載の論理装置（１０）を試験する試験方法。

【請求項１２】チップによって出力される結果を期待される結果と比較するための前記方法、および差が見つかったかどうかを示す信号を出力する前記方法は、出力結果を圧縮すること、および前記圧縮結果を、圧縮された前記期待される結果と比較することを含むことを特徴とする請求項１０記載の論理装置（１０）を試験する試験方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、大規模集積回路（ＶＬＳＩ）装置中で実施される複雑な組合せおよび順序の論理回路を試験する方法に関するものである。

【０００２】

【従来の技術およびその課題】ＶＬＳＩ装置のどこかで誤りが発生すると、その影響は、装置の試験可能な出力に到達するまでに多数のゲートを伝播する。レベル・センシティブ・スキャン・デザイン（ＬＳＳＤ）規則は、このような伝播により引き起こされるテストの複雑さを除去するために考案されたものである。この規則は、第１４回デザイン・オートメーション・カンファランスのプロシーディングのページ４６２－４６８に“ＬＳＩ試験可能性のための論理設計構造”と題する論文の中に、Ｅ．Ｂ．Ｅｉｃｈｅｌｂｅｒｇｅｒと、Ｔ．Ｗ．Ｗｉｌｌｉａｍｓとにより始めて示されたものであり、それによれば論理回路はクロック構造とされ、また、論理回路のすべての入力および出力を共に接続して一連のシフトレジスタ・スキャンパスが形成される。図１にこのような原理を用いて構成した簡単なテスタを示す。テストユニット４０は、テストパターン・データを格納するためのメモリ４２と、被試験装置の応答をシミュレートするための手段４４と、被試験装置にテストパターンを供給するための手段４６とを備えている。ユニット４０で発

3

生される試験信号は従って被試験装置１０に、シフトレジスタを形成するために共に接続された装置の入力３０を用いて与えられる。入力値は、図中の点線で示される接続を用いてユニット４０からシフトレジスタを通じてクロックに同期して入力される。入力値をクロックに同期して被試験論理回路１５を通過させることによって試験が行われた後、結果は、シフトレジスタを形成するためにやはり共に接続された出力ラッチ２０に現れる。これらの出力値はクロックに同期してシフトレジスタからコンパレータ５０に出力される。このコンパレータは、得られた結果をシミュレーションにより期待される結果と比較し、比較の結果、差があった場合には、被試験装置１０における誤りの存在を示す信号を出力する。

【０００３】論理テストパターンを発生するための従来の方法は、“確定格納パターン試験”（ＤＳＰＴ）として知られている。この方法では、確定したアルゴリズムがパターンを生成するために用いられ、このパターンはスタック・アット誤り（すなわち、ゲートがその出力を入力に応答して変化させないという誤り。ただし、検出されるのはこの誤りに限定されるものではない。）を含む特定の論理誤りの検出を保証するものである。各テストパターンおよびそれに対する期待される出力応答は、信号値の圧縮されないベクトルとしてテスタの中に格納されるので、大容量のメモリが必要となる。回路が複雑になるほど、回路中のゲートの数が増加するとスタック・アット誤りは劇的に多くなることがすでに示されている。そのことは、回路を試験するために必要なテストパターンの数も大幅に増加することを暗に意味している。従って、回路を試験するのに必要な時間が長くなり、すべての試験ベクトルを格納するためのメモリの容量も増加する。

【０００４】テストパターンを発生するために必要な時間を短縮し、発生されたパターンを格納するためのメモリの容量を減らすために、いわゆる自己試験方法が開発された。この方法では、疑似ランダムパターン発生器、および装置に実際に組み込まれた応答圧縮構造を用いる。このような構造を用い、試験に必要なエレメントを直接、被試験装置に配置することにより、テストパターンを発生するために必要なコンピュータの時間を削減することができる。この方法により、極めて多数のテストパターンを、リーズナブルな時間で装置に与えることが可能となる。Ｋｏｎｅｍａｎｎ，Ｍｕｃｈａ、ならびにＺｗｅｌｅｈｏｆｆによる２つの論文、“組み込み論理ブロック観察技術”（１９７９、ＩＥＥＥテスト・カンファランス、ページ３７－４０、Ｃｈｅｒｒｙ　Ｈｉｌｌ、ＮＪ、１９７９年１０月）および“複雑なディジタル集積回路のための組み込み試験”（ＩＥＥＥ　固体回路ジャーナル、ＳＣ－１５巻、３号、ページ３１５－３１９、１９８０年６月）には、線形帰還シフトレジスタ（ＬＦＳＲ）と呼ばれるシフトレジスタ・スキャンパス

4

の改良について開示されており、それは入力信号発生器および／または出力データ圧縮回路として用いることができよう。

【０００５】特別の自己試験アーキテクチャーがＳＴＵＭＰＳ法として知られている。ＳＴＵＭＰＳは、ＭＩＳＲ（マルチ入力シグナチャレジスタ）および並列ＳＲＳＧ（シフトレジスタ．シーケンス発生器）を用いた自己試験の略である。この方法の基本原理はよく知られており、いくつかのドキュメントに示されている（例えば、欧州特許第１０８，２５６号明細書、米国特許第５１９，０７８号明細書、米国特許第７１３，６０５号明細書、米国特許第４．９１０．７３５号明細書、あるいはＢ．Ｉ．ＤＥＲＢＩＳＯＬＧＬＵ“疑似ランダム試験のためのスキャンパス・アーキテクチャ”（ＩＥＥＥコンピュータの設計と試験、１９８９年８月、ページ３２－４８））。ＳＴＵＭＰＳによって試験データの格納容量を大幅に低減できるが、試験の質は高くない。ＢＡＳＳＥＴ他の論文“高密度論理コンポーネントのローコスト試験”（ＩＥＥＥコンピュータの設計と試験、１９９０年４月、ページ１５－２７）には、ＳＴＵＭＰＳでは、チップの代表的断面において、９０％程度の誤り検出しか期待できないと報告されている。この論文にはまた、ＳＴＵＭＰＳとＤＰＳＴとの組み合せによって、誤り検出率を高めることができると結論されている。しかし、最後の５～１０％の誤り（いわゆる自己試験エスケープ）を検出するために、経験的には、ＤＰＳＴテストパターン全体の５０～７０％のテストパターンを発生しなければならない。

【０００６】これにかわってＢＡＳＳＥＴＴ他は、重み付けランダムパターン試験（ＷＲＰＴ）を用いている。そこでは、ＬＦＳＲパターン発生器の設計は、各テストパターンの入力ビットに対する論理“１”および論理“０”の分布が可変であるように変更されている。この方法では、必要に応じて、１または０のいずれかの確率をより大きくするよう、ラッチの入力に選択的にバイアスがかけられる。誤り検出率はＳＴＵＭＰＳより大きく改善されるが、テストパターンを発生するために実質的に一層、複雑なハードウェアが必要となる。ＷＡＩＣＵＫＡＵＳＫＩによる“重み付けランダム・テストパターンを発生するための方法”と題する論文（ＩＢＭ研究開発ジャーナル、第３３巻、２号、１９８９年３月、ページ１４９－１６１）には、様々なＷＲＰＴパターンを発生する方法が示されている。しかし、得られた結果によれば、誤り検出率は９４－９９％であり、そのため、全誤りの９９．９％を検出するためには、格納されたテストパターンの５０％程度を用いて、確定パターン試験を行わなければならない。

【０００７】従来の技術で、入力パターンを発生し、同時に出力される結果を圧縮するためにＬＳＦＲ回路を用いることを開示したものはない。

5

【０００８】

【課題を解決するための手段】本発明の目的は、装置の論理を試験するために用いることができる方法およびテストパターン発生器を与えることである。テストパターンを発生するために用いられる手段は、複雑なハードウェアも個々のテストパターンを格納するための大規模のメモリスペースも必要としない。

【０００９】テストパターン発生器は直接、チップ上に組み込むことができ、また一つの装置として組み立てることもできる。チップ上に組み込まれた場合には、親子の自己試験に用いることができ、一層有用である。

【００１０】本発明はこのような目的を達成するため、簡単なＬＦＳＲを用い、その出力は被試験装置の入力を形成する一連のシフトレジスタのスキャンパスに入力する。ＬＦＳＲには最初、一連の初期値、すなわちシード、が入力され、そのシードにより被試験装置の特定の入力ラッチに必要なデータ値がクロックに同期して入力される。論理装置のすべてのゲートが確実に試験されるようにするため、すべてのシードは小さいメモリに格納される。

【００１１】

【実施例】図２は被試験装置の内部論理回路を示す図であり、これによって発明の背後の基本原理を理解できる。装置は、入力ラッチ１００ａ～ｎと出力ラッチ１４０ａ，１４０ｂとの間に接続された多数の論理ゲート１１０ａ～ｆ，１２０ａ～ｃ，１３０ａ～ｈを備えている。なお、これは単に説明のための一例であり、実際の被試験装置は、さらに多くの入力ポートおよび出力ポート、ならびにこれらのポートの間に接続されたさらに多くの論理ゲートを備えていよう。入力ラッチおよび出力ラッチはそれぞれ、共に接続されて図の点線で示すスキャンパスを形成している。ここで、装置の論理回路における入力ラッチ１００から出力ラッチ１４０に至る経路＊

6

＊について見ると、入力ラッチ１００ａ～１００ｈの値が１４０ａに出力される応答に影響することが分かる。同時に、入力ラッチ１００ｅ～１００ｎの値は出力１４０ｂにおける結果に影響する。このことは、ラッチ１００ａ～１００ｈから出力ラッチ１４０ａに至るものと、入力ラッチ１００ｅ～１００ｎから出力ラッチ１４０ｂに至る２つのコーンを定義できることを意味する。これらのコーンは、図中、破線によって示されている。また、２つのコーンはオーバーラップし、論理ゲート１２０ａ～ｃは両方のコーンに入っている。このことは、コーン１のすべての論理ゲートが試験され、誤りがなかった場合、原理的には、コーン１に含まれないコーン２のゲートのみを試験すればよく、それによって回路（すなわち、ゲート１３０ａ～ｈ）の試験は完了することを意味する。さらに、コーン２を試験するとき、試験された論理ゲート１２０ｃの出力は、試験されていない論理ゲート１３０ｆの入力に接続されているので、ラッチ１００ｅ～ｈに設定されるビットの値を無視することはできない。しかし、ラッチ１００ａ～ｄに設定されたビットの値は、コーン２のラッチの試験には関係がない。このことは、論理ゲートを試験するために、３種類の異なるタイプのビットが定義でき、ラッチ１００ａ～ｎに設定されるということを意味する。すなわち、試験関連ビット位置（表１にはＲで示す）であり、コーンの試験のために特定の値を取らなければならないものと、サポートネット、ビット位置（表１にはＡで示す）であり、コーンの試験のために複数の特定値のうちの一つをとらなければならないものと、無関係ビット位置（表１にはＸで示す）であり、コーンの試験には重要でないものとの３種類である。表１に、図２の論理ネットワークを完全に試験するために必要な３つの異なる試験手順を示す。

【００１２】

表１

ラッチ１００への	コーン１	重複部	コーン２
ネットワーク入力	ａｂｃｄ	ｅｆｇｈ	ｉｊｋｌｍｎ
手順　Ａ）：	コーン１およびコーン２に共通にネットワークを試験する		
	ＡＡＡＡ	ＲＲＲＲ	ＡＡＡＡＡＡ
	.....	.....	.....
手順　Ｂ）：	コーン１およびコーン２のネットワークを試験する		
	ＲＲＲＲ	ＡＡＡＡ	ＲＲＲＲＲＲ
	.....	.....	.....
手順　Ｃ）：	コーン２のネットワークを試験する		
	ＸＸＸＸ	ＡＡＡＡ	ＲＲＲＲＲＲ
	.....	.....	.....

フラグの意味：

Ｒ：試験関連ビット位置　－　選択不可

Ａ：サポートネット入力　－　選択可

Ｘ：無関係ビット位置　－　任意の値

手順Ａでは、ラッチ１００ｅ～ｈの値のみが、コーン重複部の論理ゲート１２０ａ～ｃの試験のために重要であ

7

る。試験手順においては、１と０のすべての組み合せがラッチ１００ｅ～ｈに設定され、論理ゲート１２０ａ～ｃはすべての条件のもとで試験される。ラッチ１００ａ～ｄに格納されるビットとラッチ１００ｉ～ｎに格納されるビットとは、出力ゲート１４０ａ，１４０ｂで意味のある出力が得られなければならないという点でのみ関係がある。手順Ｂでは、個々のコーンのゲートが試験されるが、重複部のゲートは試験されない（すなわち、ゲート１１０ａ～ｆおよび１３０ａ～ｈは試験され、ゲート１２０ａ～ｃは試験されない）。このことは、ラッチ１００ａ～ｄ，１００ｉ～ｎに設定されたビットの値は、試験に関連していて、１と０とのすべての可能な組み合せを通して循環されるが、ラッチ１００ｅ～ｈはサポートネット入力であることを意味する。最後に手順Ｃでは、コーン２の論理ゲート１３０ａ～ｈだけが試験される。ラッチ１００ｊ～ｎはこの場合にも試験に関連しており、すべての可能な組み合せが試験されるが、ラッチ１００ｆ～ｈのビットはサポートネットビットである。しかし、この場合にはラッチ１００ａ～ｄのビットの値は試験結果に無関係である。従って、装置の論理ゲートの試験に用いられる大多数のテストパターンは、冗長であることが分かる。なぜなら、それらは回路の誤りに関して何も新しい情報を生まないからである。

【００１３】試験関連ビットのアイデアは、ＬＦＳＲにおいて論理回路を試験するためのテストパターンを発生させるために利用できる。これをどのように行うかを示すため、まず最初にＬＦＳＲを用いたパターン発生法についてレビューする。図３に簡略化したＬＦＳＲ２００を示す。このＬＦＳＲは、スキャンパス２１０をクロックに同期して伝送される出力２０５を生成する３つのラッチ２００ａ～ｃを備えている。ラッチ２００ａの出力はＸＯＲゲート２０２およびラッチ２００ｂの両方に接続されている。ラッチ２００ｃの出力は、出力２０５に（従って、スキャンパス２１０に）、そしてＸＯＲゲート２０２にも接続されている。データがＬＦＳＲをクロックに同期して伝送されるとき（２００ａの値が２００ｂに配置され、２００ｂの値が２００ｃに配置され、２００ｃの値がスキャンパスに伝送されることを意味する）、２００ａの新しい値は、２００ａの以前の値と２００ｃの以前の値とに対するＸＯＲ操作の結果となる。ＬＦＳＲ２００のラッチ２００ａ～ｃに対して適切な初期値を選択することによって、スキャンパスにおいて可能な範囲でいかなるビットの組み合せも発生できる。図３に、ＬＦＳＲ２００にロードされた初期値１００がいかにスキャンパス２１０において一連の０と１を発生するかを示す。

【００１４】ここで、図３のような疑似ランダム・テストパターンを発生する代りに、スキャンパスのラッチ２１０のいくつかに格納されるビットが固定であるテストパターンを発生させるとする。これらのビットは上述した試験関連ビット位置を表す。図４に、ラッチ３１０ｃ，３１０ｅに格納されるビットがそれぞれ１および０にセットされるシミュレーション・スキャンパス３１０を示す。他のラッチに格納されるビットの値は無関係である。計算のため、ＬＦＳＲ３００のラッチ３００ａ～ｃの値が任意にそれぞれａ，ｂ，ｃとセットされたとすると、ビットをシミュレーション・スキャンパス３１０からＬＦＳＲ３００にクロックバックさせてＬＦＳＲ３００に対する最初のシード値を得ることができ、その値を後で、必要な値を持つ試験関連ビット位置によって必要なテストパターンを発生させるために用いることができる。シミュレーションとして行うこのクロックバックの手順を図４に種々のステップにより示す。なお、クロックバックは実際の装置で起こるものではない。

8

【００１５】最初のステップでは、ラッチ３１０ｂ～ｇに格納された値はすべて１ラッチ分左にシフトされる。ラッチ３１０ｇに以前格納されていた値は現在はラッチ３１０ｆに格納されており、ラッチ３１０ｆの値は今は３１０ｅに格納されている。他のラッチについても同様である。ラッチ３１０ａに格納されていた値は試験に関連しないビットであるため、それは無視され、従って、ラッチ３００ｃは、ラッチ３００ａ，３００ｂに以前格納されていた値に対するゲート３０２によるＸＯＲ操作の結果を受け取ることになる。実際、この段階でスキャンパス３１０からＬＦＳＲ３００には試験関連ビット値はクロックバックされないので、ラッチ３００ｃに格納された値は純粋に任意であり、簡単のため、それを“ａ”と定義する。次のステップでは、ラッチ３１０ｂ～ｇに格納された値は再び１ラッチ分左に移動される。ラッチ３１０ａに格納された値は今は試験関連ビットであり、その値は１である。それはＬＦＳＲ３００のラッチ３００ｃに送られる。このステップでラッチ３００ｃに格納されるべき値は、ＸＯＲゲート３０２で行われるＸＯＲ操作の結果であり、すなわちｂ＋ｃである。従って、このＸＯＲ操作の結果は１であると定義できる。ステップ３およびステップ４では、ラッチ３１０ａに格納された値は、試験に関連しないビットであり、そのため、ラッチ３００ｃに格納された値はいなかる値もとる必要がなく、従ってその値はラッチ３００ａ，３００ｂに格納された値に対するＸＯＲ操作の結果である。ステップ５ではしかし、ラッチ３００ｃにはラッチ３１０ａからビット０が入力される。このことは、ラッチ３００ａは以前は値１を格納し、ラッチ３００ｂは値ａ＋ｂを格納していたので、ＸＯＲ操作の結果を１＋（ａ＋ｃ）に定義する。最後の２つの値は試験に関連しないビットであるが、シミュレーションはさらに２つのステップ、すなわちステップ６，７と続く。しかし、これらのステップは、ＬＦＳＲが完全なテストパターンを発生するとき、ＬＦＳＲが、スキャンパス３１０のラッチ３１０ａ～ｇに対して既知の値のビットを生成すること、および

9

正しい位置に試験関連ビットを生成することを保証するために必要である。この操作の結果は最後のステップ７によって知ることができ、ラッチ３１０ａは０を、ラッチ３００ｂは値ａを、ラッチ３００ｃはａ＋１を格納しているはずである。ａは値０または１をとるので、このことは２つのシード（００１あるいは０１０）が必要なテストパターンを発生することを意味する。図５に、シードを０１０とした場合のテストパターンの発生を示す。

【００１６】もちろん実際の試験では、スキャンパスが７つのラッチだけしか含まず、その２つの値だけが試験に関連し、従ってそれらの値が固定されているということはない。しかしＬＦＳＲの規模を拡大し、ＸＯＲゲートに接続するためのフィードバック経路を適切に選ぶことによって、必要とするいかなるパターンも発生することが可能となる。ＬＦＳＲに与えるシードだけを格納すればよいので、このテストパターン発生システムを実現するために必要なメモリの容量は、決められたすべてのパターンを直接格納する場合より、大幅に小さいものとなる。

【００１７】以上説明したテストパターン発生器は、チップに直接組み込んで、装置の迅速な自己試験を行えるようにしてもよく、また、試験装置に組み込んで、被試験装置をそれに取り付けるようにしてもよい。メモリのスペースをさらに減らし、試験処理を高速化するには、出力される試験結果を、Ｊ．Ｌ．ＣＡＲＴＥＲによる論文“ＶＬＳＩ回路のための改良されたシグナチャ試験”（ＩＢＭ技術開示公報、第２６巻、３Ａ号、１９８３年８月、ページ９６５－９６７）に示されているよく知られたマルチ入力シグナチャ・レジスタ（ＭＩＳＲ）などの方法によってさらに圧縮すればよい。

10

【００１８】

【発明の効果】本発明により、複雑なハードウェアも個々のテストパターンを格納するための大規模のメモリスペースも必要としない論理回路試験方法およびテストパターン発生器が得られる。

【図面の簡単な説明】

【図１】装置を試験する基本的な方法を示す図である。

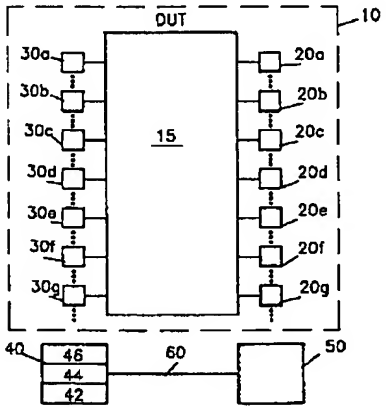
【図２】被試験装置の一例を示す図であり、入力ゲートと出力ゲートとの間に接続された論理ゲートを示す。

【図３】簡単な線形フィードバック・シフトレジスタを示す図である。

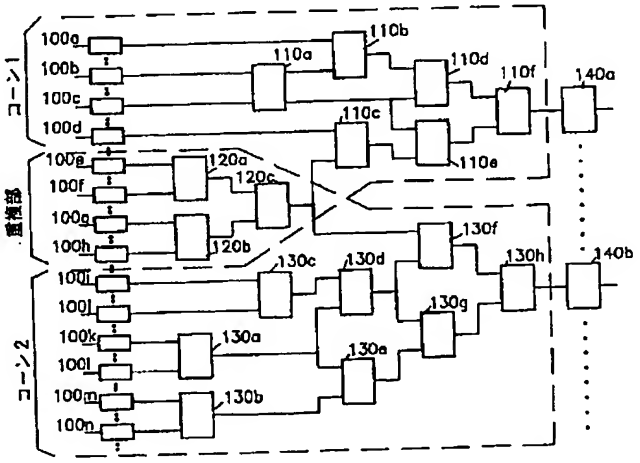
【図４】シードの発生の一例を示す図である。

【図５】シードの発生の一例を示す図である。

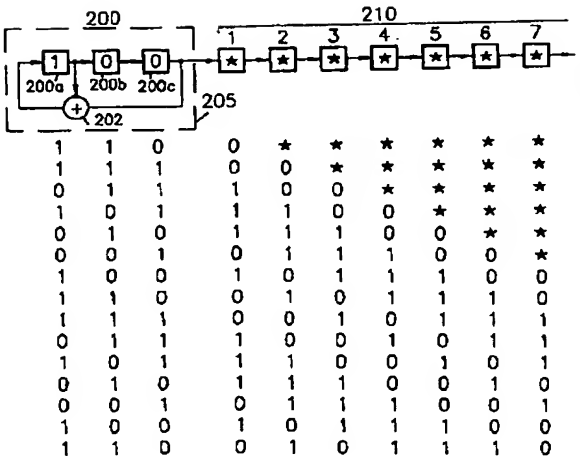
【図１】



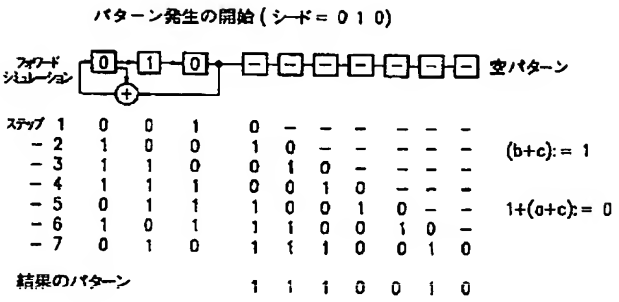
【図２】



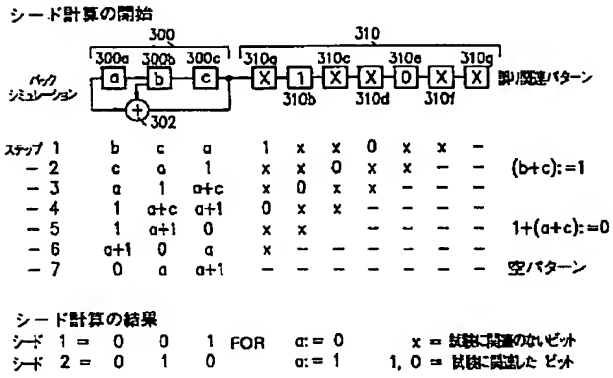
【図３】



【図５】



【図４】



フロントページの続き

(72)発明者　ヨアチム・リーグレル
ドイツ連邦共和国　7268　ゲチンゲン　バインガルテンシユトラーセ　9
(72)発明者　ペテル・ロスト
ドイツ連邦共和国　7260　ガルプ　インデン　ルンケタイレン（番地なし）
(72)発明者　マンフレツド・シユミツト
ドイツ連邦共和国　7036　シエーナイツヒダーヘンヘウゼルベーク　37
(72)発明者　オツト・トルライテル
ドイツ連邦共和国　7022　ラインヘルデンーエヒテルデインゲン　フラインシユパツフアシユトラーセ　14
(72)発明者　ペテル・ベルベーゲン
ドイツ連邦共和国　7030　ベブリンゲンホーヘンシユタウヘン　シユトラーセ　11
(72)発明者　ダウン・バイランド
ドイツ連邦共和国　7277　ビルトベルク　4　イン　ハイネンテル　21
(72)発明者　デイーテル・ベンデル
ドイツ連邦共和国　7032　シンデルフインゲン　シエーネベルゲル　ベーク　14